Vol.2016-MPS-111 No.22
2016/12/12

アセンブラ命令の出現順序とその頻度に基づいた バッファオーバーフロー攻撃の検知

南後吉秀[†1]　松田健[†2]　園田道夫[†3]　趙晋輝[†4]

概要：バッファオーバーフロー攻撃はシステムの機能停止やあふれたデータの実行をもたらす攻撃であり，確保したメモリ領域に対して許容範囲を超えるデータを送るときに発生する．既知の攻撃パターンに基づいて検知を行うシグネチャマッチング方式などの多くの対策手法が今までに研究されているが，既存の対策手法を回避する攻撃が開発され，さらに組込みソフトウエアで新たなリスクが生じる可能性もあるため，未知の攻撃への対応策の検討は重要な課題である．本研究では，シェルコードをはじめとしたアセンブラ命令列を，各命令を中心に前後複数個の命令の出現順序に着目する近傍の概念を取り入れた頻度解析を行い，その結果をもとにコサイン尺度を用いて差異を求めることによって，バッファオーバーフロー攻撃を検知する手法を提案する．

キーワード：バッファオーバーフロー，シェルコード，アセンブラ命令，出現順序，出現頻度

Detection of Buffer Overflow Attack based on Appearance Order of Assembler Instructions and their Frequency

YOSHIHIDE NANGO[†1]　TAKESHI MATSUDA[†2]
MICHIO SONODA[†3]　JINHUI CHAO[†4]

1. はじめに

近年，インターネットの急速な普及により，老若男女問わずインターネットが使われる時代となった．しかしながらその一方で，ネットワーク上でのサイバー攻撃も年々増加し続ける状況下にある．特に，システムに被害を与える不正アクセスの原因の一つとされているのが，古典的な攻撃であるバッファオーバーフロー攻撃と呼ばれるものである[1]．バッファオーバーフロー攻撃対策における既存手法としてシグネチャマッチング方式がよく知られているが，既知の攻撃の検知しか行うことができず，その上すべての攻撃の検知が原理的に不可能である．

そこで本研究では，インテル x86 アーキテクチャ環境下でバッファオーバーフロー攻撃を引き起こすシェルコードにおけるアセンブラ命令の出現状況に着目し，アセンブラ命令で記述されたシェルコードにおけるアセンブラ命令の出現ペアを出現順序と出現頻度の観点から解析することによって，バッファオーバーフロー攻撃に用いられるシェルコードの特徴抽出とバッファオーバーフロー攻撃の検知について検討及び考察する．

†1 中央大学大学院理工学研究科情報工学専攻
Department of Information and Systems Engineering, Graduate School of Science and Engineering, Chuo University
†2 長崎県立大学情報システム学部情報セキュリティ学科
Department of Information Security, Faculty of Information Systems, University of Nagasaki
†3 サイバー大学 IT 総合学部 IT 総合学科
Department of Comprehensive Information Technology, Faculty of Comprehensive Information Technology, Cyber University
†4 中央大学理工学部情報工学科
Department of Information and Systems Engineering, Faculty of Science and Engineering, Chuo University

2. バッファオーバーフローとシェルコード

2.1 バッファオーバーフロー

バッファオーバーフロー（バッファオーバーラン）とはよく知られているセキュリティホールの一種である．これによる脆弱性はコンピューターの黎明期から現在に至るまで見かけることができ，Internet Explorer をはじめとするプログラムで発見されたゼロデイ脆弱性でもバッファオーバーフローが用いられている[2]．バッファオーバーフロー攻撃はバッファオーバーフローを利用した攻撃であり，一例としてサービス妨害攻撃と呼ばれる DoS 攻撃（Denial of Service Attack）や，DoS 攻撃にとってかわった DDoS 攻撃（Distributed Denial of Service Attack）が存在する．バッファオーバーフロー攻撃による被害の事例として，2000 年 1 月から 2 月にかけて発生した中央省庁 Web サイトの改ざんが知られている[1]．

2.2 シェルコード

シェルコードはシェルを起動してコマンドを受け付けるようにするプログラムである．コンピューターセキュリティの分野におけるシェルコードは，ソフトウエアの脆弱性を利用するペイロード（パケットの中でヘッダーを除いた部分）として使われるコード断片であり，バッファオーバーフローを効果的に利用するためにシステムに送られるものでもある．シェルコードをシステムに応じてカスタマイズすることによって，脆弱性を抱えたプログラムから制御を奪い取ることができる．これによって行われる操作の一例として，コンピューターに管理者アカウントを追加したり，ログファイルから記録を抹消したりすることが挙げら

れる[2]．シェルコードは機械語の命令列で構成されたコードであることから，16 進数やそれに対応するアセンブラ命令で記述されていることが多い．

3. 既存手法

3.1 シグネチャマッチング方式

この方式は，バッファオーバーフロー攻撃対策における既存手法としてよく知られており，侵入検知システム（Intrusion Detection System, IDS）において用いられていることが多い．例えば IDS としてよく知られている Snort の場合では，セキュリティホールに対する攻撃パケットの固有な部分と攻撃の根幹を成すシェルコードと呼ばれる部分の特徴的な部分に関わるものをシグネチャと呼ばれるデータベースに登録しておき，シグネチャの内容とネットワーク上を流れるパケットの内容とを比較することによって攻撃検知を行っている．しかしこの方式においては，未知のセキュリティホールの検出ができず，さらにシェルコードのパターンが無限に作成可能であるがゆえにすべての攻撃の検出が原理的に不可能である，ということが欠点として存在する[3]．

4. 提案手法

4.1 着目対象とするアセンブラ命令

本研究を進めるにあたり，Linux におけるバッファオーバーフロー攻撃で用いられるシェルコードを対象として，攻撃コードで頻繁に用いられているインテル x86 アーキテクチャのアセンブラ命令について，従来研究[4]で調査している．この調査で用いたシェルコードは，シェルコートデータベースとして知られている shell-storm [5] に存在するシェルコードデータ 100 個である．これらのシェルコードデータに対する調査の結果，攻撃コードで頻繁に用いられるアセンブラ命令は次の 6 つであり，本研究でもそれを踏襲する．ここでは[6]における説明を簡潔に記す．

- INT 命令（Call to Interrupt Procedure）
  割り込み処理を呼び出すためのアセンブラ命令である．
- LEA 命令（Load Effective Address)
  実効アドレス（制御装置が命令を実行する時に実際に使用するアドレス）をロードするためのアセンブラ命令である．
- MOV 命令（Move）
  データを転送するためのアセンブラ命令である．
- POP 命令（Pop a Value from the Stack）
  スタック領域からデータを取り出すためのアセンブラ命令である．
- PUSH 命令（Push Word of Doubleword onto the Stack）
  スタック領域へデータを挿入するためのアセンブラ命令である．
- XOR 命令（Logical Exclusive OR）
  排他的論理和の演算を行うためのアセンブラ命令である．

ここで，オペコードとはマイクロプロセッサーなどに与える機械語の命令の識別番号を表す．

4.2 アセンブラ命令ペアの頻度解析

本研究においては，シェルコードデータ中の基準となるアセンブラ命令（基準命令）を先頭から最終行まで 1 行ずつ変化させて，各基準命令における前後 K 個のアセンブラ命令（近傍命令）について，近傍命令から基準命令への遷移のペア，もしくは基準命令から近傍命令への遷移のペアを解析対象として数える．例として，図 1 に示すシェルコードデータについて考えることとする．第一に，K=2 として基準行が 3 行目（push から始まる行）である場合を考えると，解析対象となるアセンブラ命令ペアは，nop→push，xor→push，push→lea，push→mov，の 4 つである．第二に，K=2 として基準行が 5 行目（mov から始まる行）である場合を考えると，push→mov，lea→mov，mov→int の 3 つが解析対象のアセンブラ命令ペアとなる．

```
nop
xor     ecx,ecx
push    ecx
lea     eax,[ecx+0Bh]
mov     ebx,esp
int     80h
```

図 1　シェルコードデータの一例

4.3 出現頻度ベクトルの導入

アセンブラ命令ペアの出現頻度解析で得られた結果をベクトル化して考える．ここで，攻撃検知に用いるデータにおけるアセンブラ命令ペア出現頻度を集約したベクトルを **i** とし，攻撃データにおけるアセンブラ命令ペア出現頻度を集約したベクトルを **a** とし，正常データにおけるアセンブラ命令ペア出現頻度を集約したベクトルを **b** とする．また，以降これらのベクトルを出現頻度ベクトルと呼ぶこととし，これらの出現頻度ベクトルの次元数 n は着目対象とするアセンブラ命令の数を asm として asm+1 の 2 乗とする．4.1 節より，本研究では asm = 6 として議論を進める．

4.4 出現頻度ベクトルの内積とコサイン尺度の算出

次に，ベクトル化して得られた出現頻度ベクトルに対して，内積 **i・a**，**i・b** とコサイン尺度 cos(**i**, **a**)，cos(**i**, **b**)を算出する．これらは次の 4 つの式で算出できる．

$$\mathrm{i}\cdot\mathrm{a} = \sum_{j=1}^{n} i_j a_j = i_1 a_1 + i_2 a_2 + \cdots + i_n a_n$$

$$\cos(\mathrm{i},\mathrm{a}) = \frac{\mathrm{i}\cdot\mathrm{a}}{\sqrt{i_1^2 + i_2^2 + \cdots + i_n^2}\sqrt{a_1^2 + a_2^2 + \cdots + a_n^2}}$$

$$\mathrm{i}\cdot\mathrm{b}=\sum_{j=1}^{n} i_j b_j = i_1 b_1 + i_2 b_2 + \cdots + i_n b_n$$

$$\cos(\mathrm{i},\mathrm{b}) = \frac{\mathrm{i}\cdot\mathrm{b}}{\sqrt{i_1^2 + i_2^2 + \cdots + i_n^2}\sqrt{b_1^2 + b_2^2 + \cdots + b_n^2}}$$

4.5 類似度に基づいた攻撃検知

本研究においては，内積をベースとした類似度をもとに攻撃検知を行うこととしている．しかし，ベクトルには長さの概念が存在するため，内積そのものを用いて攻撃検知を行おうとすると誤検知の発生が考えられる．そのため，n 次元の座標空間上に **i**，**a**，**b** が存在しているものとみなして，**i** と **a** のなす角と **i** と **b** のなす角，つまりコサイン尺度 cos(**i**, **a**)，cos(**i**, **b**)を比較して攻撃検知を行うこととする．攻撃検知に用いるデータが攻撃データか正常データかどうかについては，なす角を基準として **i** が **a** と **b** のどちらに近いかという観点で判定する．つまり，条件式で記述すれば以下の通りである．

cos(**i**, **a**)＞cos(**i**, **b**)：攻撃データ

cos(**i**, **a**)≦cos(**i**, **b**)：正常データ

このような判定基準になる根拠として，$0 \leqq \theta \leqq \pi$ において θ が大きくなるほど $\cos\theta$ の値が小さくなるためである．

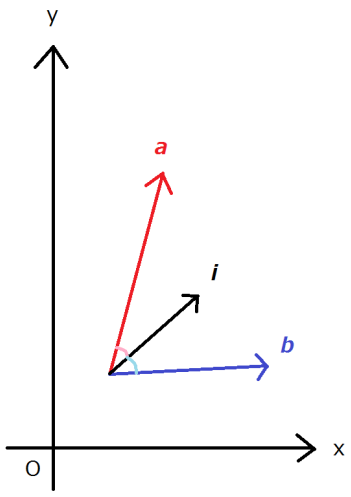

図 2　xy 平面をモデルとした攻撃検知のイメージ

5. 実装結果

5.1 攻撃データでのアセンブラ命令ペア頻度解析の結果

ここでは，4.2 節で示した頻度解析を攻撃データに対して行った結果を示す．ここで示した結果は出現頻度ベクトル **a** の要素として使用する．頻度解析に使用したデータは，シェルコードデータベースの shell-storm に存在するシェルコードデータ 200 個である．表 1 に示す頻度解析の結果は，解析対象のデータ 200 個全体におけるアセンブラ命令ペアの出現頻度をペアごとに集計し，それらの出現頻度の合計を基準として算出したものである．また，表 1 における行方向は遷移元命令，列方向は遷移先命令を表している．

表 1　攻撃データにおける頻度解析の結果（単位は%）

	INT	LEA	MOV	POP	PUSH	XOR	他
INT	0.20	0.05	1.81	0.32	1.57	1.48	2.22
LEA	0.33	0.17	0.38	0.00	0.21	0.14	0.15
MOV	4.84	0.68	5.93	0.53	6.51	2.52	2.48
POP	0.89	0.04	0.64	0.38	1.28	0.63	1.03
PUSH	2.07	0.19	10.33	2.12	21.01	0.92	1.78
XOR	1.00	0.08	2.78	0.31	3.62	1.69	1.29
他	1.08	0.17	2.07	0.69	3.01	1.36	5.05

5.2 正常データでのアセンブラ命令ペア頻度解析の結果

次に，4.2 節で示した頻度解析を正常データに対して行った結果を示す．ここで示した結果は出現頻度ベクトル **b** の要素として使用する．頻度解析に使用したデータは表 2 に示す 200 個である．表 3 に示す頻度解析の結果は 5.1 節と同様に，解析対象のデータ 200 個全体におけるアセンブラ命令ペアの出現頻度をペアごとに集計し，それらの出現頻度の合計を基準として算出したものである．また，表 3 における行方向は遷移元命令，列方向は遷移先命令を表している．

表 2　頻度解析に使用した正常データとその個数

データの説明	個数
自作の C 言語プログラムを gcc コマンドによりコンパイルしたもの	30 個
柴田望洋氏の C 言語サンプルプログラム[7]を gcc コマンドによりコンパイルしたもの	30 個
ネットワーク通信を含む C 言語プログラム[8]を gcc コマンドによりコンパイルしたもの	40 個
Vine Linux 6.3 環境下における/bin/sh を objdump コマンドにより逆アセンブルしたもの	50 個
Vine Linux 6.3 環境下における Linux コマンドを objdump コマンドにより逆アセンブルしたもの	50 個

表 3　正常データにおける頻度解析の結果（単位は%）

	INT	LEA	MOV	POP	PUSH	XOR	他
INT	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LEA	0.00	0.45	2.83	0.01	0.21	0.03	1.27
MOV	0.00	2.14	23.21	0.28	0.44	0.22	17.59
POP	0.00	0.13	0.18	0.76	0.04	0.01	0.96
PUSH	0.00	0.01	0.80	0.05	0.99	0.03	2.44
XOR	0.00	0.03	0.28	0.06	0.00	0.02	0.25
他	0.00	2.10	16.46	0.94	2.36	0.34	22.06

6. 考察

表 1 で示したアセンブラ命令ペアの頻度解析結果より，頻度解析に用いた攻撃データ 200 個について，最も多く出現したアセンブラ命令ペアは PUSH 命令→PUSH 命令のペアであり，全く出現しなかったアセンブラ命令ペアは LEA 命令→POP 命令のペアであった．スタックにおけるバッファオーバーフロー攻撃ではすぐに処理しきれないほどの大量のデータをスタック領域に挿入して，プログラムに意図しない動作を実行させるようにする．データをスタック領域に挿入するアセンブラ命令が PUSH 命令であることも踏まえた上で，PUSH 命令→PUSH 命令のペアが多数出現した要因になったと考えられる．

一方で，表 3 で示したアセンブラ命令ペアの頻度解析結果より，頻度解析に用いた正常データ 200 個について，最も多く出現したアセンブラ命令ペアは MOV 命令→MOV 命令のペアであり，全く出現しなかったアセンブラ命令ペアは INT 命令の関係するペア（遷移元命令と遷移先命令のうち少なくとも一方が INT 命令であるペア）であった．本研究で用いた正常データには，標準入力から読み込まれた値やそれをもとに計算された値を変数等へ代入するような箇所が多数存在していた．代入を行うために用いられるアセンブラ命令が MOV 命令であることも踏まえた上で，MOV 命令→MOV 命令のペアが多数出現した要因になったと考えられる．また，正常データの頻度解析結果では，着目対象外命令同士の遷移のペアが MOV 命令→MOV 命令のペアの次に出現する結果となった．これは，頻度解析に用いた正常データにおいて，特定のアドレスへ移動するための JMP 命令や戻り先のアドレスを記憶して特定のアドレスへ移動するための CALL 命令が出現していたことが要因として考えられ，これらのアセンブラ命令を着目対象命令として取り入れることで攻撃特徴の変化が起こりうる可能性が存在する．

従来研究[9]においては，アセンブラ命令で記述されたコードについて，アセンブラ命令の出現順序の特定と機能面での分類分けにより特徴を見出し，マルコフ連鎖モデルと SVM（Support Vector Machine）を用いてシェルコードの検知を行っている．一方で本研究においては，アセンブラ命令のペアを考えることでアセンブラ命令の出現順序に着目しつつ，アセンブラ命令で記述されたコードにおけるアセンブラ命令ペアの出現頻度を数えることによって特徴を見出し，特徴をまとめた出現頻度ベクトルをもとにコサイン尺度を算出および比較して攻撃検知を行うものである．本研究では，着目対象とするアセンブラ命令を 4.1 節に示した 6 つに限定しているが，着目対象を見直すことでより正確に攻撃の特徴を見出せるのではないかと考えられる．

7. まとめと今後の課題

本研究では，3.1 節で説明した未知のパターンの攻撃検知が原理的に不可能であるというシグネチャマッチング方式の欠点を補うことを目的として，特定のアセンブラ命令に着目してそれらのアセンブラ命令の出現順序をペアとして考え，近傍の概念を取り入れて各アセンブラ命令ペアの出現頻度を数えることで攻撃の特徴を見出し，類似度を表す指標の一種であるコサイン尺度を用いて攻撃検知を行う手法について提案した．アセンブラ命令ペアの出現頻度解析の結果，攻撃データ 200 個中では PUSH 命令同士の遷移のペアが全体の 5 分の 1 強を占めており，一方で正常データ 200 個中では MOV 命令同士の遷移のペアが全体の 4 分の 1 弱を占めていた．

今後の課題として，第一に攻撃検知のテストと交差検証を行い攻撃検知結果の妥当性を検証すること，第二にアセンブラ命令ペアの出現頻度解析に用いた正常データについて，妥当性を検証して必要に応じて使用するデータの見直しを検討すること，第三に着目対象とするアセンブラ命令について，追加や変更による見直しを検討すること，が挙げられる．

参考文献

[1] WEB マーケティング研究会，“トラブル事例に学ぶ Web サイトのセキュリティ 第 5 回・バッファオーバーフロー攻撃”，http://www.webdbm.jp/column2009/column2009-03/2330/，2016 年 11 月 11 日閲覧．
[2] Jon Erickson 著，村上 雅章 訳，“Hacking 美しき策謀 脆弱性攻撃の理論と実際 第 2 版”，オライリー・ジャパン，2011.
[3] 北條 孝佳，佐久間 英夫，種茂 文之，“シェルコード解析による不正アクセス検出手法”，情報処理学会研究報告 2003-CSEC-23，2003.
[4] 南後 吉秀，松田 健，園田 道夫，趙 晋輝，“16 進数コードの出現状況に着目したバッファオーバーフロー攻撃の特徴抽出”，第 14 回情報科学技術フォーラム A-019，2015.
[5] Jonathan Salwan，“Shellcodes database for study cases”，http://shell-storm.org/shellcode/，2016 年 11 月 11 日閲覧．
[6] インテル株式会社，Intel Corporation，“IA-32 インテル アーキテクチャ・ソフトウェア・デベロッパーズ・マニュアル”, 2004.
[7] 柴田望洋後援会，“柴田望洋博士の著書＋翻訳書”，http://www.bohyoh.com/Books/index.html，2016 年 11 月 11 日閲覧．
[8] 小俣 光之，種田 元樹，“「Linux ネットワークプログラミングバイブル」サポートページ”，http://www.ncad.co.jp/~mtaneda/lnpb/, 2016 年 11 月 11 日閲覧．
[9] Ziming Zhao，Gail-Joon Ahn，“Using Instruction Sequence Abstraction for Shellcode Detection and Attribution”，2013 IEEE Conference on Communications and Network Security．
[10] Krerk Piromsopa，Richard J. Enbody，“Buffer-Overflow Protection: The Theory”，2006 IEEE International Conference on Electro/Information Technology．
[11] Desheng Fu，Feiyue Shi，“Buffer Overflow Exploit and Defensive Techniques”，2012 Fourth International Conference on Multimedia Information Networking and Security．
[12] 愛甲 健二, “たのしいバイナリの歩き方”, 技術評論社, 2013.